REMARKS

It is believed that the above amendments and following remarks attend to each and every rejection and objection presented in the August 24, 2004 Office Action. Claims 7 and 19 are amended, respectively, for antecedent basis and a typographical error, and without adding new matter. Claims 1-19 remain pending, with claims 1 and 14 being independent.

Specification

The Examiner objects to the title as not being descriptive of the invention to which the claims are directed. The title is amended hereinabove. Reconsideration is requested.

Claim Objections

The Examiner objected to antecedent basis of "switch event" in claim 7. Claim 7 is thus amended to provide clear antecedent basis. Reconsideration is requested.

Claim Rejections – 35 U.S.C. §102

Claims 1-19 stand rejected under 35 U.S.C. §102(b) as being anticipated by Concurrent Event Handling through Multithreading, 1999, IEEE Transactions on Computers, volume 48, NO. 9, pages 903-916 (hereinafter "Keckler"). Respectfully, we disagree. To anticipate a claim, Keckler must teach every element of the claim and "the identical invention must be shown in as complete detail as contained in the ... claim." *MPEP 2131* citing *Verdegaal Bros. V. Union Oil Co. of California*, 814 F.2d 628, 2 USPQ2d 1051 (Fed. Cir. 1987) and *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 9 USPQ2d 1913 (Fed. Cir. 1989). Keckler does not teach every element of claims 1-19.

Claim 1 recites a method for determining thread switch points within pipeline execution units of a processor, including:

a) monitoring instruction processing of a first thread within the pipeline execution units;

b) in the event of a possible switch point within the pipeline execution units, deactivate the first thread, or not, based upon a first urgency indicator for the first thread.

An 'urgency' for the thread is for example based upon "how well a thread is progressing (or will be progressing) within a pipeline" and "how urgent the program or processor logic believes the thread should be". See Applicant's specification, paragraph [0007]. "An assessment is also made of a thread's execution and relative to timeslice expiration." See Applicant's specificaiton, paragraph [0009].

Keckler, on the other hand, discloses a processing chip that "makes its thread selections based upon the availability of an instruction's operands." See Keckler page 908, section 3.2. Keckler also does not disclose or suggest an urgency indicator upon which deactivation of a thread is based. Teaching away from the invention of claim 1, Keckler instead teaches a "simple three state priority scheme" that "enables critical threads ... to use a higher fraction of the execution resources..." See Keckler, page 908, section 3.2.1. The priority system as described by Keckler is thereby static, and depends upon the application of the thread (i.e., a user thread, a system thread, etc).

Keckler does not therefore disclose or suggest 'deactivating the first thread, or not, based upon a first urgency indicator for the first thread' as required by claim 1. Reconsideration of claim 1 is requested.

Claims 2-13 depend from claim 1 and benefit from like arguments; but in addition these claims have other features that patentably distinguish over Keckler. For example, Keckler does not disclose or suggest using a second urgency indicator of a second thread to decide upon deactivating the first thread and activating the second thread, as required by claim 2.

Keckler also does not disclose deactivating the second thread, or not, based upon the second urgency indicator for the second thread and in the event of a possible switch point event of the second thread, as required by claim 3.

Keckler also does not disclose or suggest activating another thread within the pipeline if the second thread is switched out, as in claim 4.

Moreover, Keckler does not teach or suggest modifying an urgency indicator based upon associated possible switch point and timeslice expiration, as required by claims 7 and 8.

Keckler further has no teaching or suggestion of modifying a thread's urgency indicator by inserting an instruction to the pipeline, as required by claim 11. The Examiner argues that Keckler teaches a time slice of 255 cycles, page 908. We respectfully disagree: Keckler clearly points out that 'a ready instruction can only be stalled for up to 255 cycles.' This is not the same as timeslicing as in the immediate application. Keckler's multithreading is implemented by interleaving instructions from different threads over execution resources of a cluster on a cycle-by-cycle bases.

Reconsideration of claims 2-13 is requested.

Claim 14 recites a processor for processing multi-threaded program instructions, including:

i. an array of pipeline execution units and associated heuristics affecting how the instructions are processed within the units; and

ii. a thread controller for monitoring processing of the instructions within the units and for switching between multiple program threads based upon (a) the heuristics and (b) urgencies of the program threads.

Keckler does not disclose or suggest a thread controller for monitoring processing of the instructions within pipeline execution units and for switching between multiple program threads based upon heuristics and urgencies of the program threads, as required by element ii of claim 14. Instead, and as noted above, multithreading is implemented by interleaving instructions from different threads over the execution resources of each cluster on a cycle-by-cycle bases. Keckler specifically teaches away from claim 14 by reciting that "unlike block multithreading [1], [34], which switches threads on long latency operations, ... the MAP chip makes its thread selections based upon the availability of an instruction's register operands." See Keckler page 908, section 3.2.

By way of comparison, multithreading within the immediate application switches threads based upon its 'urgency', which in turn may be based upon "a thread instruction missing the cache" or "processor interrupts" (see paragraphs [0006] and [0007]).

Reconsideration of claim 14 is requested.

Claims 15-19 depend from claim 14 and benefit from like argument; but in addition, these claims also have features that are patentably distinct from Keckler. For

example, Keckler does not disclose or suggest timeslice heuristics as in claim 15. Keckler does not disclose or suggest program threads with one of the instructions changing urgency of at least one thread of the processor, as in claim 16. Keckler further does not disclose or describe a controller modifying an urgency of any of the threads to modify future treatment of the threads in switch out events, as in claim 17. Keckler also does not describe the controller decreasing or increasing the urgency for the program threads by injecting an instruction into the pipeline execution units, as in claim 18. Finally, it is clear that Keckler also does not disclose or suggest a time slice expiration unit for monitoring expiration of threads within the processor, as in claim 19.

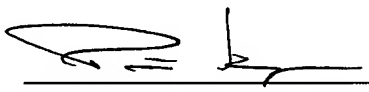
Reconsideration of claims 15-19 is requested.

In view of the above arguments, we contend that claims 1-19 patentably distinguish over Keckler. Generally, Keckler describes only how to select which thread becomes active at an event. In accord with the inventions of claims 1-19, on the other hand, active and inactive events may be examined and may or may not switch based on determination of urgency.

Claims 1-19 are therefore deemed allowable. Should the Examiner disagree, we ask for the opportunity to interview.

It is believed that no fees are due in connection with this amendment. If any additional fee is due, please charge Deposit Account No. 08-2025.

Respectfully submitted,

By: 

Peter C. Knops, Reg. No. 37,659
LATHROP & GAGE L.C.
2345 Grand Blvd., Suite 2400
Kansas City, Missouri 64108
Telephone: (816) 460-5826
Facsimile: (816) 292-2001